DOCUMENTAZIONE PROGETTO DI RETI INFORMATICHE 2022/2023

A cura di Francesco Taverna

Scelte relative all'implementazione del Server

Il server è iterativo e utilizza I/O multiplexing per gestire l'input, le prenotazioni dei client, le comande relative ai vari Table Device e la loro gestione da parte dei Kitchen Device.

Motivazione utilizzo I/O multiplexing e server iterativo: una limitazione che si presenta nell'utilizzo di un server iterativo è la gestione di un'unica richiesta alla volta da parte da parte dello stesso. Tuttavia, questa scelta si può considerare adeguata ed efficiente tenendo conto di alcuni fattori, tra cui la limitatezza del codice da eseguire, l'ottima gestione delle richieste garantita dall'I/O multiplexing attraverso il controllo dei socket pronti e dello stdin, ed il ristretto numero di device connessi contemporaneamente. Ciononostante, potrebbe accadere che, nei momenti di maggiore attività, si possa perdere qualche richiesta avendo previsto una coda di sole 24 posizioni.

Devices

I programmi client si occupano di ricevere comandi da tastiera, svolgere la gestione relativa al comando scelto ed effettuare eventuali controlli sul suo formato e sulla sua tipologia.

Tipologia di Devices:

1. **Client**

Il client non necessita di un I/O multiplexing come gli altri device perché non deve ricevere nessuna notifica, ma solamente informazioni relative ai tavoli disponibili. Un tavolo può essere mostrato solo se il numero di persone è uguale alla capienza del tavolo oppure la differenza tra la capienza del tavolo e il numero di persone desiderato è <= 3. Inoltre, ogni tavolo si considera prenotato per 2 ore massimo, poi torna ad essere libero. Se si effettuasse una find e, prima di fare la book, il tavolo pensato nella scelta non fosse più disponibile, verrà mostrato uno specifico messaggio di errore. La prenotazione è effettuabile entro l'orario in cui il ristorante risulta aperto (dalle 9:00 alle 23:00) e solo ad orari "interi" (9,10,11, ...).

2. **Table Device**

I Table Device sono strutturati per ricevere inizialmente una stringa contenente il codice di prenotazione precedentemente ottenuto tramite il programma client e fintantoché non viene inserito un codice corretto, le loro funzionalità risultano bloccate. Questo programma prevede l'utilizzo dell'I/O multiplexing dato che deve ricevere informazioni aggiornate sullo stato attuale delle comande da esso inviate.

Si è ipotizzato che qualsiasi codice permetta di accedere ad un tavolo, anche se il codice non appartiene propriamente a quel tavolo (circostanza che non si dovrebbe presentare in quanto i clienti verranno accompagnati al tavolo corretto). L'accettazione di un qualsiasi codice è resa possibile attraverso la comunicazione con il server dopo il login, il quale controlla a che tavolo è associato quel codice (nella prenotazione si salva anche la coppia id_tavolo-codice). Nonostante ciò, nel codice del table device, sono presenti dei commenti che permettono di settare la variabile globale NUMERO_TABLE_DEVICE tramite il parametro posto nello script di esecuzione e, di conseguenza, identificare se il table device nel quale si sta immettendo il codice sia precisamente quello associato al codice di prenotazione, in caso negativo si mostra a video lo specifico errore. [Commenti righe: 88, 162-166, 178-184 del td1.c]. La richiesta del conto è permessa anche se non si ordina nulla, in quel caso restituirà 0.

3. Kitchen Device

Il Kitchen Device, come il Table Device, è gestito con I/O multiplexing per essere aggiornato in tempo reale sulla presenza di comande in attesa di accettazione. Ogni KD non può accettare più di 10 comande; con "*comanda accettata"* si intende una comanda che è ancora *"in preparazione"* e non ancora "*in servizio"*. Il numero di Kitchen Device può essere gestito tramite la macro *NUMERO_KITCHEN_DEVICE* oppure tramite il parametro posto nello script che verrà stampato a video. Il comando *set* necessita di uno spazio anziché di un **"–"** tra id_comanda e tavolo (scelta adottata per semplicità).

Server

Per gestire la comunicazione con i vari device si salvano i file descriptor nei corrispondenti array. Se l'utente ha immesso "stop", il numero 1024 verrà inviato, se possibile, a tutti i Kitchen Device e Table Device connessi, andando ad identificare il segnale di chiusura e disconnessione. Il server inizialmente riceve un intero per discriminare la tipologia di device che lo ha contattato: 0-9 Kitchen Device, 10-79 client, 80-99 table device. Il server stampa i dati sulla prenotazione ricevuti dal client. Il codice di prenotazione risulta la concatenazione dell'id del tavolo scelto, della data e dell'ora per garantire univocità. Il server gestisce tramite l'array "*arrivate*" di 400 elementi le comande ricevute, si ipotizza un caso peggiore di 20 comande a tavolo per una unica fascia oraria, ogni elemento dell'array mi permette di salvare anche lo status attuale della comanda e presenta un attributo che ne controlla la validità. Questa scelta presenta una criticità: se si arrivasse a 400, l'indice verrebbe resettato ma quando un Kitchen Device accetta una comanda non accetterà la più recente. Si potrebbe risolvere salvando il timestamp della comanda ed effettuando una ricerca nell'array con timestamp più datato. Il server si occupa di segnalare l'accettazione delle comande da parte dei KD verso i TD, oltre che le notifiche relative alle comande arrivate dai TD verso i KD. Per le prenotazioni si utilizza un array di 1000 elementi per gestirne il grande numero, anche nel lungo periodo. Nel server si utilizzano invio di interi e costrutto switch per il riconoscimento dei comandi digitati nei device e si esegue il codice relativo al comando digitato.

Scelte sui Socket

Sono stati scelti socket del tipo:

- TCP: si utilizza il TCP per garantire affidabilità di comunicazione, dato che la perdita di dati in questo contesto peggiorerebbe notevolmente la qualità del servizio e potrebbe causare disagi gravi al ristorante. Inoltre, data la bassa stima di messaggi, si presuppone che non ci siano rallentamenti tramite l'utilizzo di questo protocollo, di conseguenza l'UDP non porterebbe vantaggi
- Bloccante: usato per la dipendenza delle operazioni rispetto alle precedenti. Infatti, se si volesse usare un socket non bloccante sorgerebbero problemi legati alla mancanza di dati necessari che devono essere ottenuti dalla controparte

Salvataggio in memoria

Si è preferito il salvataggio in memoria rispetto al salvataggio in file per semplicità di gestione, utilizzando array è risultato più facile modificare le strutture.

Protocollo di comunicazione

Si utilizza un protocollo *text* per inviare le stringhe e un protocollo *binary* per inviare gli interi. Quando si invia una stringa si invia un intero per definirne la lunghezza e si mette il carattere di fine stringa solamente nella ricezione.

Nella cartella si può trovare la topologia del ristorante.